Chapter 25
Development of Design and Training Application for Deep Convolutional Neural Networks and Support Vector Machines

Fusaomi Nagata, Kenta Tokuno, Akimasa Otsuka, Hiroaki Ochi, Takeshi Ikeda, Keigo Watanabe, and Maki K. Habib

Abbreviations

AlexNet	AlexNet is the name of a well-known convolutional neural network designed by Alex Krizhevsky, which is the champion of the competition called ImageNet Large Scale Visual Recognition Challenge held in 2012
Back Propagation (BP) algorithm	BP algorithm is a famous method used in artificial neural networks to calculate weights between a large number of neurons. The BP algorithm is a generalized delta rule for multi layered neural networks, in which chain rules are applied to iteratively calculating the weights based on gradients and errors in the network

F. Nagata (✉) · K. Tokuno · A. Otsuka · H. Ochi · T. Ikeda
Sanyo-Onoda City University, Sanyo-onoda, Japan
e-mail: nagata@rs.socu.ac.jp; otsuka_a@rs.socu.ac.jp; ochi@rs.socu.ac.jp; t-ikeda@rs.socu.ac.jp

K. Watanabe
Okayama University, Okayama, Japan
e-mail: watanabe@sys.okayama-u.ac.jp

M. K. Habib
The American University in Cairo, Cairo, Egypt
e-mail: maki@aucegypt.edu

O. Sergiyenko et al. (eds.), *Machine Vision and Navigation*,
https://doi.org/10.1007/978-3-030-22587-2_25

Caffe	Caffe means convolutional architecture for fast feature embedding, which is one of famous deep learning frameworks developed at University of California, Berkeley. Caffe is an open source library written in C++
Convolutional Neural Network (CNN)	A convolutional neural network called CNN is a class of deep neural networks, which has been most commonly applied to image recognition. The CNNs are typical applications based on the concept of deep learning and are known as one of the most powerful structures for image recognition
MATLAB	MATLAB is a high performance computing environment provided by MathWorks. In particular, MATLAB enables us to conduct matrix manipulations, plotting of data, implementation of algorithms, design of user interfaces, and development of application software written in standard languages such as C++, C#, Python, and so on
ReLU	ReLU means a rectified linear unit function. In the structure of recent deep neural networks, the ReLU is one of effective activation functions superior to conventional sigmoid functions, which is defined with only positive part of the input argument
Sequential Minimal Optimization (SMO) Algorithm	SMO is an effective algorithm to solve quadratic programming (QP) problems. The QP is one of nonlinear programming methods. The SMO was proposed by John Platt in 1998 and has been used to train support vector machines
Support Vector Machine (SVM)	In machine learning, the support vector machine called SVM is one of supervised learning models associated with learning algorithms which can analyze training data used for classification and regression problems. A binary class SVM model is a hyper plane that can clearly separate two categorized points in space. The hyper plane is drawn so that the points can be divided into two domains by a clear margin as large as possible
TensorFlow	TensorFlow is an open source software library which can be used for the development of machine learning software such as neural networks. TensorFlow was developed by Google and released from 2015

25.1 Introduction

Recently, deep learning techniques are gathering attention from researchers and engineers all over the world due to the high performance superior to conventional shallow neural networks. In this decade, several software development environments for deep neural networks (DNN) such as Caffe [1] and TensorFlow [2] have been introduced to researchers and engineers. In those development environments, C++ or Python is well used for development. Deep convolutional neural networks (DCNN) are typical applications based on the concept of DNN and are known as one of the most powerful structures for image recognition. However, for example, it may be difficult for students and junior engineers to develop and implement a practical DCNN using programming languages such as C++ or Python and to utilize it for anomaly detection in actual production systems. Generally speaking, it seems that the availability of user-friendly software that facilitates such applications without using programming languages skills, such as C++ or Python, have not been sufficiently developed yet.

Hence, this paper presents the development of user-friendly application development environment based on MATLAB system [3, 4] that facilitates two applications using DCNNs and support vector machines (SVMs). An application of DCNN for anomaly detection is developed and trained using many images to inspect undesirable defects such as crack, burr, protrusion, chipping, spot, and fracture phenomena which appear in the manufacturing process of resin molded articles. Automation of visual inspection process has been demanded from many different kinds of industrial fields because it is not easy to reduce the increase of undesirable human error associated with the length of successive working hours.

Besides DCNN, SVMs are supervised learning models with associated learning algorithms that analyze data sets used for classification and regression analysis. Not only have a linear classification ability based on the idea of margin maximized hyperplanes, but also SVMs have promising characteristics to efficiently perform a nonlinear classification using what is called the kernel trick, by implicitly mapping input data into high-dimensional feature spaces [5].

In the fields of measurement systems, for example, Flores-Fuentes et al. proposed a combined application of power spectrum centroid and SVMs to improve the measurement capability in optical scanning systems [6]. The energy signal center is found in the power spectrum centroid, in which the SVM regression method is used as a digital rectified to increase measurement accuracy for optical scanning system. Then, a technical research of an opto-mechanical system for 3D measurement was reported in detail, in which a multivariate outlier analysis was implemented to detect and remove atypical values, in order to improve the accuracy of artificial intelligence regression algorithms [7]. Also, although the architecture is not deep structure, Rodriguez-Quinonez et al. surveyed the dominant laser scanner technologies, gave a detailed description of their 3D laser scanner, and adjusted their measurement error by a once trained feed forward back propagation (FFBP) neural network with a Widrow-Hoff weight/bias learning function [8]. Surface measurement systems (SMS) allow accurate measurements of surface geometry for three-dimensional computational models creation. There are cases where contact avoidance is needed; these techniques are known as non-contact surface measurement techniques. To perform non-contact surface measurements, there are different operating modes and technologies, such as lasers, digital cameras, and integration of both. Each SMS is classified by its operation mode to get the data, so it can be divided into three basic groups: point-based techniques, line-based techniques, and area-based techniques. Real et al. provided useful topics about the different types of non-contact surface measurement techniques, theory, basic equations, system implementation, actual research topics, engineering applications, and future trends [9]. The description seems to be particularly beneficial for students, teachers, researchers, and engineers who want to implement some visual inspection system.

In this paper, two kinds of SVMs are, respectively, incorporated with the two trained DCNNs to classify sample images with high recognition rate into accept as OK or reject as NG categories, in which compressed features obtained from the DCNNs are used as the input for the SVMs. The two types of DCNNs used for generating feature vectors are our designed sssNet and the well-known AlexNet [10, 11]. The designed applications of the SVMs and their evaluation are introduced. The usability and operability of the proposed design and training application for DCNNs and SVMs are demonstrated and evaluated through design, training, and classification experiments.

25.2 Design and Training Application for DCNNs and SVMs

A large number of image files with different kinds of defect features and their paired labels for classification are needed to construct a reliable DCNN-based anomaly detection system with generalization ability. To deal with this serious need, a dialogue-based application named the similar image generator was first developed that can easily produce a lot of similar images with sequential number from an original image for training. For example, similar images with fracture defect can be generated as shown in Fig. 25.1 by rotating, translating, scaling an original image, changing the brightness, the resolution, or the file format such as JPG, BMP, PNG, and so on.

Then, a DCNN and SVM design application as shown in Fig. 25.2 was developed using App Designer provided by MATLAB. Deep Learning Toolbox (Neural Network Toolbox), Statistics and Machine Learning Toolbox, Parallel Computing Toolbox, Computer Vision System Toolbox, and Image Processing Toolbox were optionally installed for the development on MATLAB. Main DCNN design parameters on number of layers, filter size, pooling size, padding size, and width of stride can be easily given through the user-friendly dialogue. As an example, Fig. 25.3 shows a designed DCNN composed of three convolution layers. The first layer is positioned for input images with a resolution of $200\times200\times1$ given by a matrix

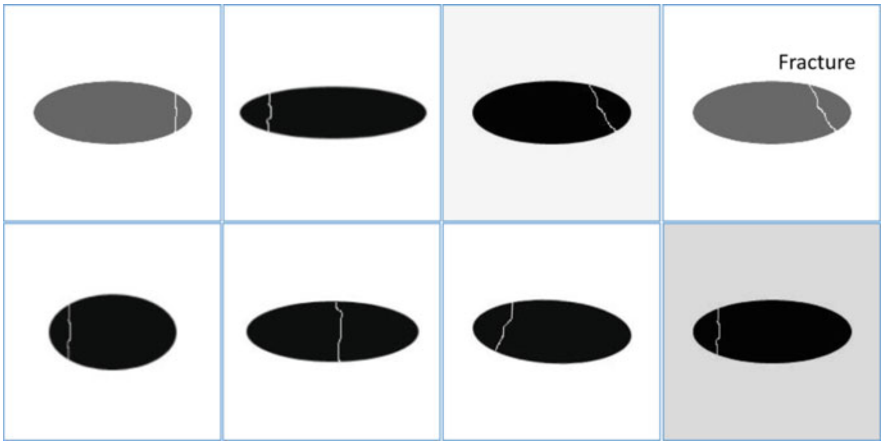
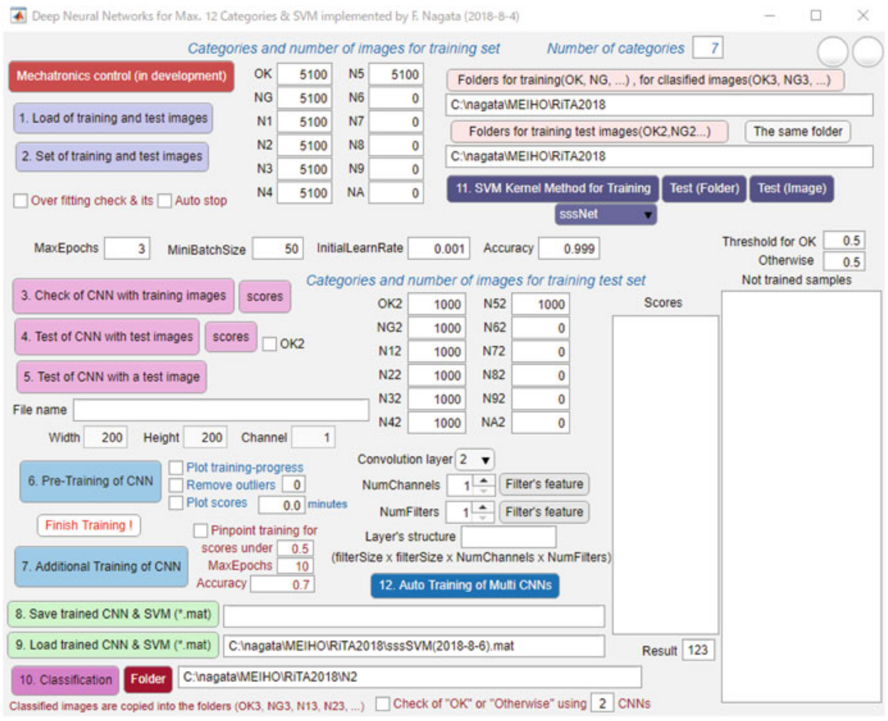


Fig. 25.1 Examples of generated images with a defect of fracture for training



Deep Neural Networks for Max. 12 Categories & SVM implemented by F. Nagata (2018-8-4)

Categories and number of images for training set — Number of categories 7

Mechatronics control (in development)

1. Load of training and test images

2. Set of training and test images

Over fitting check & its ☐ Auto stop ☐

OK	5100	N5	5100
NG	5100	N6	0
N1	5100	N7	0
N2	5100	N8	0
N3	5100	N9	0
N4	5100	NA	0

Folders for training(OK, NG, ...) , for cllasified images(OK3, NG3, ...)

C:\nagata\MEIHO\RiTA2018

Folders for training test images(OK2,NG2...) — The same folder

C:\nagata\MEIHO\RiTA2018

11. SVM Kernel Method for Training — Test (Folder) — Test (Image)

sssNet

MaxEpochs 3 — MiniBatchSize 50 — InitialLearnRate 0.001 — Accuracy 0.999

Threshold for OK 0.5

Otherwise 0.5

Not trained samples

3. Check of CNN with training images — scores

4. Test of CNN with test images — scores — ☐ OK2

5. Test of CNN with a test image

Categories and number of images for training test set

OK2	1000	N52	1000
NG2	1000	N62	0
N12	1000	N72	0
N22	1000	N82	0
N32	1000	N92	0
N42	1000	NA2	0

Scores

File name

Width 200 — Height 200 — Channel 1

6. Pre-Training of CNN

☐ Plot training-progress

☐ Remove outliers 0

☐ Plot scores 0.0 minutes

Finish Training !

☐ Pinpoint training for scores under 0.5

MaxEpochs 10

Accuracy 0.7

7. Additional Training of CNN

Convolution layer 2

NumChannels 1 — Filter's feature

NumFilters 1 — Filter's feature

Layer's structure

(filterSize x filterSize x NumChannels x NumFilters)

12. Auto Training of Multi CNNs

8. Save trained CNN & SVM (*.mat)

9. Load trained CNN & SVM (*.mat) — C:\nagata\MEIHO\RiTA2018\sssSVM(2018-8-6).mat

Result 123

10. Classification — Folder — C:\nagata\MEIHO\RiTA2018\N2

Classified images are copied into the folders (OK3, NG3, N13, N23, ...)

☐ Check of "OK" or "Otherwise" using 2 CNNs

Fig. 25.2 The developed design and training application for DCNNs and SVMs

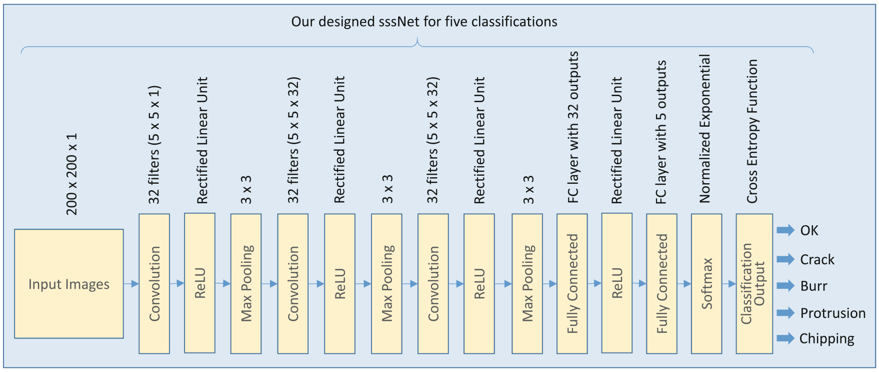


Fig. 25.3 An example of DCNN for five classifications designed by using the application shown in Fig. 25.2

with zerocenter normalization. The second, fifth, and eighth layers are convolution ones which severally have 32 filters. It is known that convolution layers perform the translation invariance and compositionality required for computer vision. In the convolution layers, the filters are applied to each image while sliding from the left top to the right bottom in the image based on the value of the stride. Note that each filter has channels as many as the number of feature maps in the previous layer. Activation functions called rectified linear unit (ReLU) are located at third, sixth, ninth, and twelfth layers. The ReLUs are given by

$$f(u) = \max(0, u) \tag{25.1}$$

$$f'(u) = \begin{cases} 1 & (u > 0) \\ 0 & (u \leq 0) \end{cases} \tag{25.2}$$

In the context of deep neural networks, ReLU have been actively used as one of the most effective activation functions for back propagation algorithms. The 4th, 7th, and 10th layers are max pooling ones to reduce the dimensions of feature maps for computational efficiency. The sizes of pooling, stride, and padding are given as [3 3], [2 2], and [0 0 0 0], respectively. If the nth image for training is given to the input layer, then the 14th softmax layer produces the probability $p_{ni}(i = 1, 2, \ldots, 5)$ called the score for five categories. The probability $p_{ni}(i = 1, 2, \ldots, 5)$ generated from the 14th softmax layer for five categories is calculated by

$$p_{ni} = \frac{e^{y_{ni}}}{\sum_{k=1}^{5} e^{y_{nk}}} \tag{25.3}$$

where $\boldsymbol{y}_n = [y_{n1}\ y_{n2}\ y_{n3}\ y_{n4}\ y_{n5}]^T$ is the output vector from the 13th fully connected layer corresponding to the nth input image. In this case, the loss function called cross entropy is calculated by

$$\bar{E} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{5} t_{nk}\log(y_{nk}) \tag{25.4}$$

where $\boldsymbol{t}_n = [t_{n1}\ t_{n2}\ t_{n3}\ t_{n4}\ t_{n5}]^T$ means the nth desired output vector for five categories, i.e., $[1\ 0\ 0\ 0\ 0]^T$, $[0\ 1\ 0\ 0\ 0]^T$, $[0\ 0\ 1\ 0\ 0]^T$, $[0\ 0\ 0\ 1\ 0]^T$, or $[0\ 0\ 0\ 0\ 1]^T$. N is the total number of image samples in the training set. The cross entropy is also used to tune the values in each filter in back propagation algorithm during iterative training process.

25.3 Review of Back Propagation Algorithm for Implementation

The authors of this chapter have implemented the back propagation algorithm into some systems [12–15]. The first system designed for a feedforward force controller learned the contact motion which was the relation between the contact force and the velocity at the tip of robot arm[12]. The second system named the effective stiffness estimator was developed for a desktop NC machine tool with a compliant motion capability. The estimator finally allowed the machine tool to generate a desired damping needed for a stable force control system without undesirable large overshoots and oscillations [13, 14]. Further, the third system was considered to deal with the problem concerning the learning performance to large scale teaching signals, so that a simple and adaptive learning technique for sigmoid functions could be proposed. The validity and control effectiveness of the learning technique were verified through simulation experiments using the dynamic model of PUMA560 manipulator with six degrees of freedoms. In this section, the important back propagation (BP) algorithm for training is reviewed for an easy implementation in software development using a simple three-layered neural network with two inputs and two outputs as shown in Fig. 25.4, in which a standard sigmoid function is applied as an activation function of each neuron. It is known that BP algorithms are also applied to the training of filters in CNNs. The sigmoid function and its derivative are generally given by

$$f(s) = \frac{1}{1+e^{-s}} \tag{25.5}$$

$$f'(s) = f(s)\{1 - f(s)\} \tag{25.6}$$

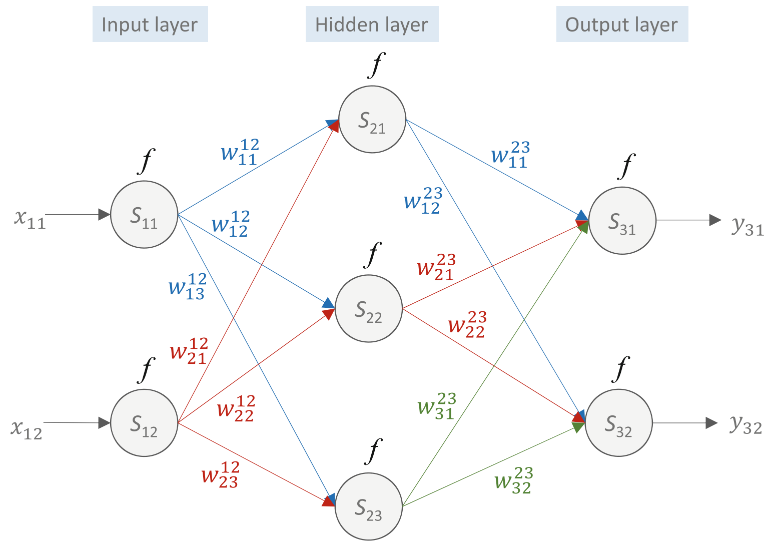


Fig. 25.4 Three-layered neural network to review the back propagation algorithm

where s is the state of a neuron. Weights between the last hidden layer and the output layer are updated through the calculations based on the generalized delta rule. The weights in this example are actually updated based on the rule as written by

$$e_{31} = d_1 - y_{31} \tag{25.7}$$

$$e_{32} = d_2 - y_{32} \tag{25.8}$$

$$w^{23}_{11} = w^{23}_{11} + \eta f(s_{21}) f(s_{31})\{1 - f(s_{31})\}e_{31} \tag{25.9}$$

$$w^{23}_{12} = w^{23}_{12} + \eta f(s_{21}) f(s_{32})\{1 - f(s_{32})\}e_{32} \tag{25.10}$$

$$w^{23}_{21} = w^{23}_{21} + \eta f(s_{22}) f(s_{31})\{1 - f(s_{31})\}e_{31} \tag{25.11}$$

$$w^{23}_{22} = w^{23}_{22} + \eta f(s_{22}) f(s_{32})\{1 - f(s_{32})\}e_{32} \tag{25.12}$$

$$w^{23}_{31} = w^{23}_{31} + \eta f(s_{23}) f(s_{31})\{1 - f(s_{31})\}e_{31} \tag{25.13}$$

$$w^{23}_{32} = w^{23}_{32} + \eta f(s_{23}) f(s_{32})\{1 - f(s_{32})\}e_{32} \tag{25.14}$$

where d_1 and d_2 are the components in the desired output vector to be trained. $\boldsymbol{y}_3 = [y_{31}\ y_{32}]^T$ is an output vector from the network. $\boldsymbol{e}_3 = [e_{31}\ e_{32}]^T$ is the error vector between the desired and the actual outputs. w^{pq}_{ij} is the weight between the ith unit in pth layer and the jth unit in qth layer. $\boldsymbol{x}_1 = [x_{11}\ x_{12}]^T$ is the input vector to be directly $\boldsymbol{s}_1 = [s_{11}\ s_{12}]^T$. η is the learning rate. Also, for example, the state s_{31} and s_{32} are linearly calculated by

$$s_{31} = w_{11}^{23} f(s_{21}) + w_{21}^{23} f(s_{22}) + w_{31}^{23} f(s_{23}) \tag{25.15}$$

$$s_{32} = w_{12}^{23} f(s_{21}) + w_{22}^{23} f(s_{22}) + w_{32}^{23} f(s_{23}) \tag{25.16}$$

Next, update process of weights between the hidden layer and the input layer is explained. The calculation of error e_{pi} for pth layer, ith unit in the hidden layer is a little bit more complex. For example, e_{21}, e_{22}, and e_{23} are obtained by

$$e_{21} = w_{11}^{23} f(s_{31})\{1 - f(s_{31})\}e_{31} + w_{12}^{23} f(s_{32})\{1 - f(s_{32})\}e_{32} \tag{25.17}$$

$$e_{22} = w_{21}^{23} f(s_{31})\{1 - f(s_{31})\}e_{31} + w_{22}^{23} f(s_{32})\{1 - f(s_{32})\}e_{32} \tag{25.18}$$

$$e_{23} = w_{31}^{23} f(s_{31})\{1 - f(s_{31})\}e_{31} + w_{32}^{23} f(s_{32})\{1 - f(s_{32})\}e_{32} \tag{25.19}$$

so that, the weight w_{11}^{12}, w_{12}^{12}, w_{13}^{12}, w_{21}^{12}, w_{22}^{12}, and w_{23}^{12} are calculated by

$$w_{11}^{12} = w_{11}^{12} + \eta f(s_{11}) f(s_{21})\{1 - f(s_{21})\}e_{21} \tag{25.20}$$

$$w_{12}^{12} = w_{12}^{12} + \eta f(s_{11}) f(s_{22})\{1 - f(s_{22})\}e_{22} \tag{25.21}$$

$$w_{13}^{12} = w_{13}^{12} + \eta f(s_{11}) f(s_{23})\{1 - f(s_{23})\}e_{23} \tag{25.22}$$

$$w_{21}^{12} = w_{21}^{12} + \eta f(s_{12}) f(s_{21})\{1 - f(s_{21})\}e_{21} \tag{25.23}$$

$$w_{22}^{12} = w_{22}^{12} + \eta f(s_{12}) f(s_{22})\{1 - f(s_{22})\}e_{22} \tag{25.24}$$

$$w_{23}^{12} = w_{23}^{12} + \eta f(s_{12}) f(s_{23})\{1 - f(s_{23})\}e_{23} \tag{25.25}$$

25.4 Design and Training Experiments of Designed DCNN

25.4.1 Test Trial of Design and Training of a DCNN for Binary Classification

Table 25.3 tabulates the main parameters of DCNN training in case of two categories, which are non-defective and defective articles named OK and NG, respectively. The category of NG includes the defects of burr, protrusion, and crack. The training was conducted by using a single PC with a Core i7 CPU and a GPU (NVIDIA GeForce GTX 1060). In this DCNN training, it first took about several minutes until the categorization accuracy reached to 0.95. The accuracy is the result of discriminant analysis obtained by dividing the number of correctly classified images by that of images in the entire data set. Then, the DCNN was additionally fine-trained to enhance the accuracy to 1. It also took several minutes to the completion. After the DCNN was trained, the classification result of the training set was checked based on the scores from the softmax layer. Figure 25.5 shows the scores of classification of OK and NG using all the images in training set (total number of images is 2040).

Table 25.3 Parameters designed for DCNN training

Filter size in convolution layers	5 × 5 × 1
Padding of convolution layers	[2 2 2 2]
Stride of convolution layers	[1 1]
Pooling size	[3 3]
Padding of max pooling layers	[0 0 0 0]
Stride of max pooling layers	[2 2]
Max epochs	30
Mini batch size	200
Learning rate	0.002 to 0.0001
Desired categorization accuracy	0.999
Number of OK images	1020
Number of NG images	1020

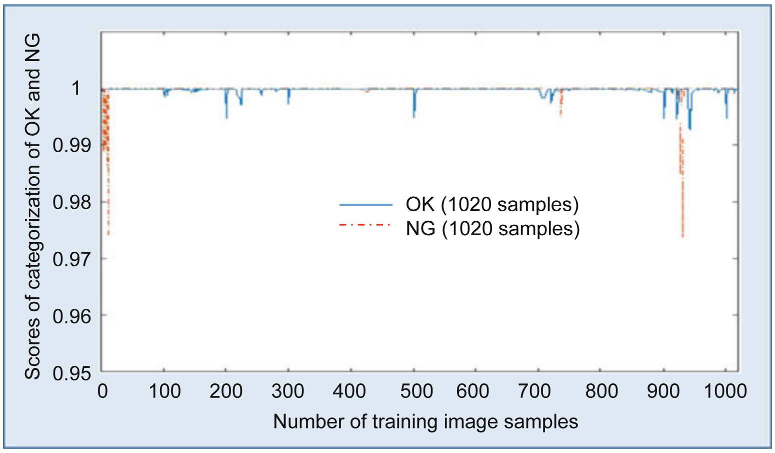


Fig. 25.5 Scores of classification of OK and NG using the all images in the training set (the total number of images = 1020 + 1020)

As can be seen, it is observed that all the 2040 images in the training set can be well discriminated with each score more than 0.97. Next, the generalization of the trained DCNN was simply evaluated using the test images with a feature of burr, protrusion, or crack as shown in Fig. 25.6 which were not included in the training set. Figure 25.7 shows the classification scores evaluated using the ten test images including a feature of defect, in which it is observed that "image2.jpg" and "image9.jpg" are not well categorized. To cope with this problem, we considered to pinpointedly improve the recognition ability to these two types of defects. Figure 25.8 shows the additional 10 training images a little bit deformed from "image2.jpg" in Fig. 25.6.

To enhance the classification ability further to the images shown in Figs. 25.6 and 25.8, the pretrained DCNN was additionally retrained using the reorganized training set consisting of original 2040 images, additional 20 OK ones, 10 NG ones in Fig. 25.6, and 10 NG ones in Fig. 25.8. After the additional training, the training

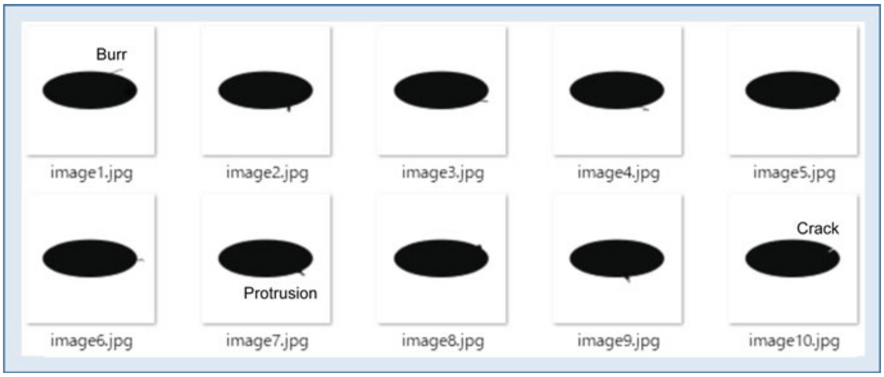


Fig. 25.6 Test images with a feature of NG which were not included in the training images

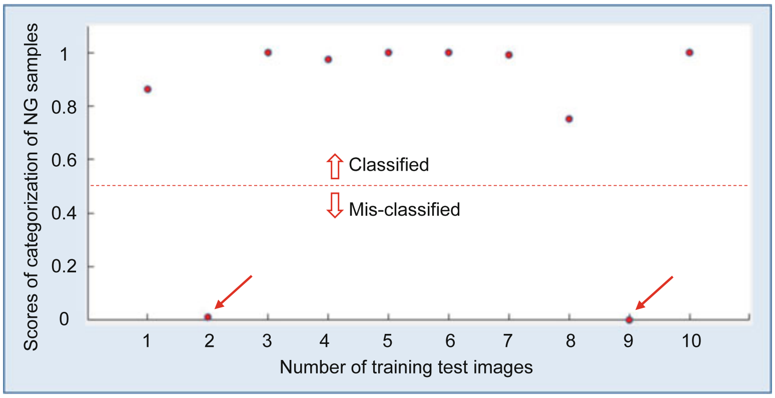


Fig. 25.7 Scores of classification of NG test images shown in Fig. 25.6

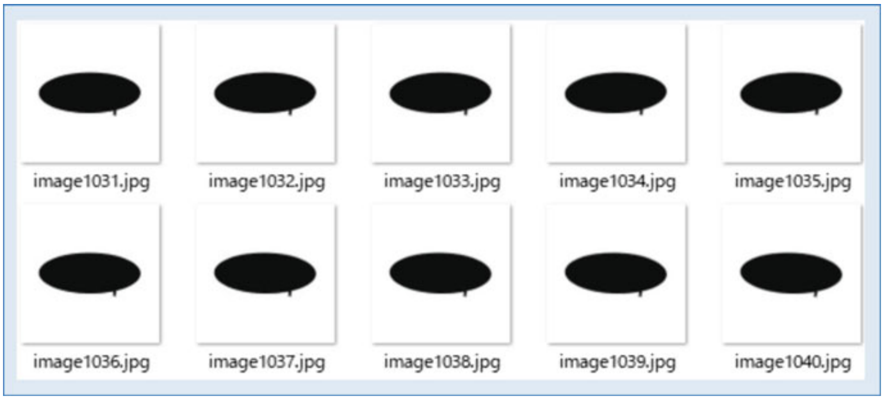


Fig. 25.8 Additional 10 training images with a protrusion which is a little bit deformed from "image2.jpg" shown in Fig. 25.6

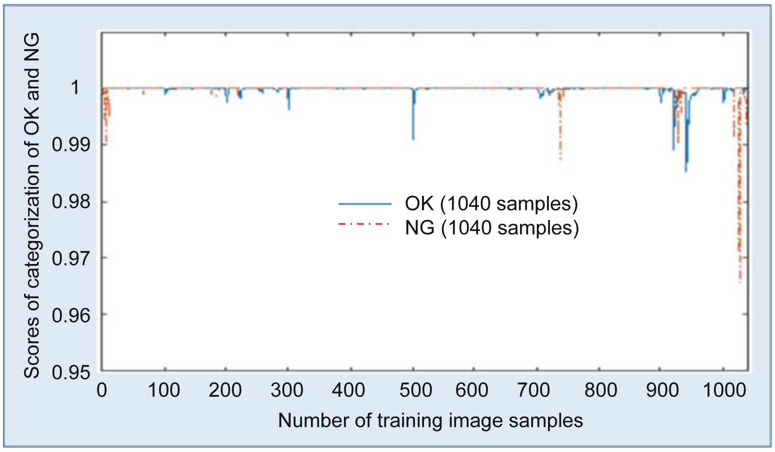


Fig. 25.9 Check of scores of categorization OK and NG using the images in training set (total number of images = 1040 + 1040), in which images shown in Figs. 25.6 and 25.8 are included

situation was checked based on the scores of categorization OK and NG using the images in reorganized training sets (total number of images = 1040 + 1040) including the images in Figs. 25.6 and 25.8. Figure 25.9 shows the result. It is observed that the recognition ability to additional images can be improved efficiently and pinpointedly. The additional training function introduced in this section is effective to reconstruct an updated DCNN when miscategorized images are found in training test process.

25.4.2 Test Trial of Design and Training for Five Categories

The DCNN designed for the binary classification of resin molded articles is extended and applied to classifying images into typical five defective categories as shown in Fig. 25.3, in which the category of NG is subdivided into typical defects seen in resin molding process such as crack, burr, protrusion, and chipping. An epoch means a full pass through the entire training data set, i.e., 5100×5 = 25,500 images are used for training process. First, a pretraining using randomly initialized weights is conducted through the period from the first epoch to sixth one, where the desired categorization accuracy is set to 0.999. Then, a fine training using the pretrained weights is successively conducted through the period from the seventh epoch to tenth one, where the desired categorization accuracy is increased to 0.9999. After the fine training, it is confirmed from the experiments that the designed DCNN with 15 layers shown in Fig. 25.3 can be well trained to classify resin molded articles into five categories, through the training process using 25,500 gray scale image samples with the resolution of 200×200.

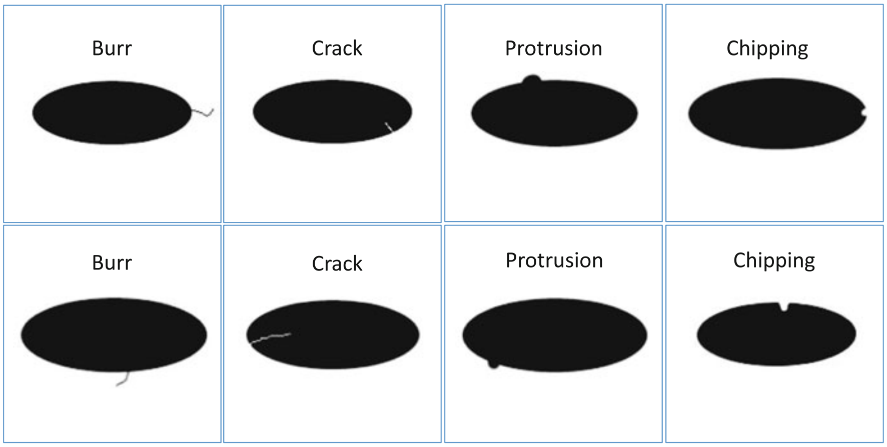


Fig. 25.10 Examples of four kinds of defects which are seen in production process of resin molded articles

Finally, after further adding 300 images with different features into each training set, the trained DCNN is additionally trained, i.e., by using the 5400×5 = 27,000 images. Then, to simply check the generalization ability of the trained DCNN, a training test set consisting of 100 images$\times$5 categories are prepared. Figure 25.10 shows some of the images in the training test set. After the testing, it is confirmed that the categorization accuracy to the test images is 492/500 = 98%, so that it is concluded that the obtained DCNN can perform satisfactory generalization.

25.5 Support Vector Machines Based on Trained DCNNs

In the previous section, two types of DCNNs for two or five classification are designed, trained, and evaluated using the proposed DCNN design application. In this section, another approach using two types of support vector machines (SVMs) is introduced. It is expected that the DCNN designed in the previous section may be able to give more characterized feature vectors to the SVMs. Actually, the most important function which is required to a defect inspection system is to remove defective products from all products. It is not allowed that any defective product is mixed into lots of non-defective products. To cope with this serious need, two types of SVMs shown at the lower parts in Figs. 25.11 and 25.12 are tried to be designed and trained using the proposed application shown in Fig. 25.2. It is expected that the trained SVMs will be able to classify input images into OK or NG category including a small defect such as crack, burr, protrusion, chipping, spot, and fracture.

As for the first SVM, our designed DCNN named sssNet is used to extract the feature vector $\boldsymbol{x} = [x_1, x_2, x_3, \ldots, x_{32}]^T$ from each inputted image. Figure 25.11 illustrates the designed SVM for binary classification whose input is the feature vector

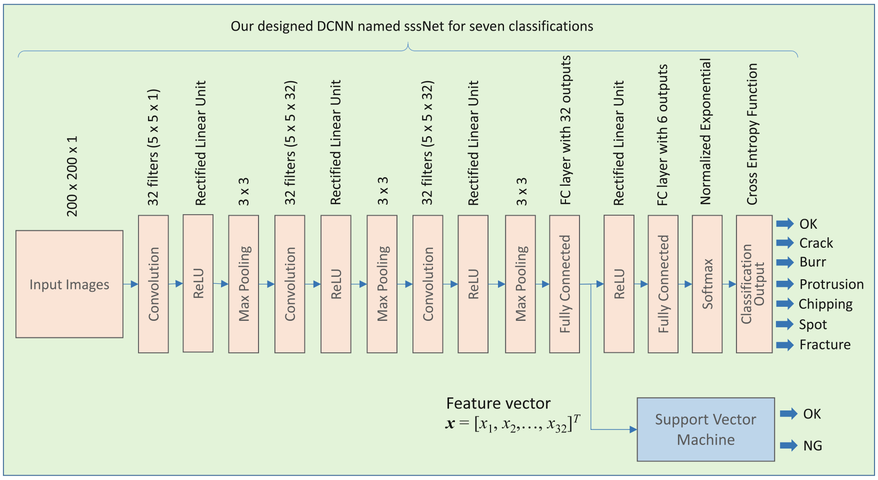


Fig. 25.11 The proposed SVM for binary classification to which feature vectors generated from our designed DCNN named sssNet are given

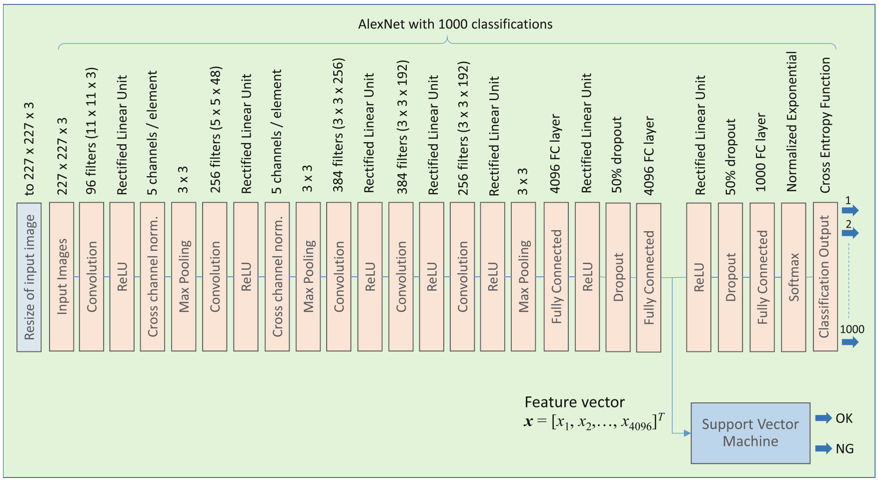


Fig. 25.12 The proposed SVM for binary classification to which feature vectors generated from AlexNet are given

generated from the 1st fully connected layer (11th layer) in the sssNet. Gaussian kernel function is used for one class unsupervised training of the SVM, in which 5100 OK images used in the pretraining in the Sect. 25.4.2 are reused. Sequential minimal optimization (SMO) algorithm [16] is applied to solve the quadratic programming (QP) of SVM. It took about several minutes for training the SVM. After training the SVM, a classification experiment was conducted to check the generalization ability to unlearned NG images. Figure 25.13 shows the classification results using

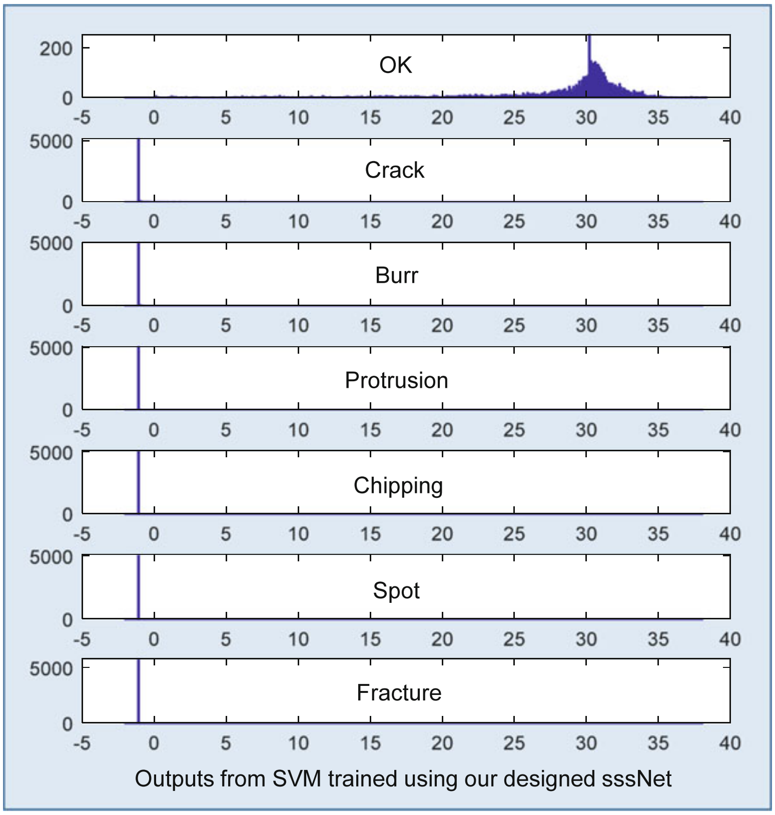


Fig. 25.13 Classification results using the SVM shown in Fig. 25.11, in which horizontal and vertical axes denote the output from the SVM trained with our designed sssNet and the number of image samples, respectively

the SVM shown in Fig. 25.11. The horizontal and vertical axes denote the output values from the SVM trained with our designed sssNet and the number of image samples, respectively. It is observed from Fig. 25.13 that the SVM can discriminate NG images from OK ones.

As for the second SVM, well-known DCNN called AlexNet is used to extract the feature vector $\boldsymbol{x} = [x_1, x_2, x_3, \ldots, x_{4096}]^T$ from each inputted image. The AleNnet trained using one million images can classify test images into 1000 object categories such as a keyboard, mug, pencil, many kinds of animals, and so on. It is known that the AlexNet learned abundant feature representations of images covering a wide range of objects. If the trained AlexNet receives an image with the resolution of 227×227×3, then the label of an object featuring in the image and the probability, i.e., score of the categorized object, are produced. Figure 25.12 illustrates another binary class SVM whose input is the feature vector generated from the 2nd fully connected layer (20th layer) in the AlexNet. Similarly, 5100 OK images

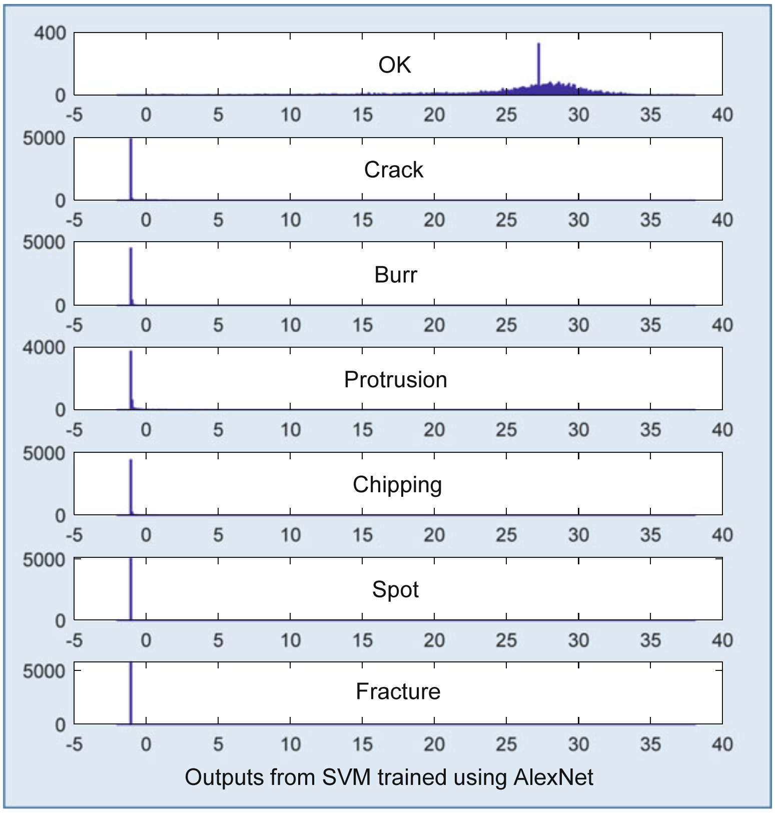


Fig. 25.14 Classification results using the SVM shown in Fig. 25.12, in which horizontal and vertical axes denotes the output from the SVM trained using AlexNet and the number of image samples, respectively

used in the pretraining in the Sect. 25.4.2 were reused for one class unsupervised training of the SVM. It also took about several minutes for training. After training the SVM, a classification experiment was conducted to check the generalization ability to unlearned NG images. Figure 25.14 shows the classification results using the SVM shown in Fig. 25.12. It is observed from Fig. 25.14 that the SVM with AlexNet can also discriminate NG images from OK ones with the almost same reliability as the SVM with sssNet. Actually, lengths of feature vectors generated from sssNet and AlexNet are quite different as 32 and 4096; however, almost the same discrimination ability can be obtained. In the case of the target features as shown in Figs. 25.1, 25.6, 25.8, and 25.10, the feature vector with 4096 components given to SVM seems to be somewhat redundant.

25.6 Conclusions

In this decade, deep learning, in particular, DCNN has been eagerly focused on by researchers and engineers in order to apply to various kinds of inspection systems. However, it seems that user-friendly design and training tools without using programming languages such as C++ and Python have not been well provided yet. In this chapter, a design and training application for DCNNs with multiple classification and SVMs with binary classification is presented. As the first application test trial, a DCNN is designed using the application to detect defects such as crack, burr, protrusion, chipping, and fracture phenomena seen in the manufacturing process of resin molded articles. A similar image generator is also proposed to efficiently generate a large number of images transformed from original ones by rotating, scaling, and changing brightness, etc. After the designed DCNN is pretrained using those images, classification experiments are conducted using test images in order to simply check the generalization. Based on the results, an additional fine training method is applied and evaluated to cope with mis-classified images, so that the classification ability can be efficiently and pinpointedly improved to a desired level of categorization accuracy. Generally, the objective of training in machine learning is to enhance the ability of generalization to unlearned environments. The additional training introduced in this chapter may proceed to the opposite direction of the objective of training or cause different kinds of problems. However, in daily production process, it will be effective for the construction of a practical visual inspection system. Here, the practicality means that the additionally trained DCNN will never miss the defects which have been misrecognized once. As the second application test trial, two kinds of SVMs with trained DCNNs, i.e., our designed sssNet and well-known AlexNet, for binary classification are designed, trained, and evaluated to discriminate NG sample images from OK ones, so that it is confirmed that the SVM with our designed sssNet can perform almost the same recognition ability as that with AlexNet in spite of the much shorter feature vector.

Finally, the authors apologize that unfortunately this chapter cannot show real photos including defective plastic parts due to the obligation of confidentiality with a joint research and development company.

References

1. Cengil, E., Cnar, A., & Ozbay, E. (2017). Image classification with caffe deep learning framework. In *Proceedings of 2017 International Conference on Computer Science and Engineering (UBMK)*, Antalya (pp. 440–444).
2. Yuan, L., Qu, Z., Zhao, Y., Zhang, H., & Nian, Q. (2017). A convolutional neural network based on tensorflow for face recognition. In *Proceedings of 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing (pp. 525–529).

3. Nagata, F., Tokuno, K., Tamano, H., Nakamura, H, Tamura, M., Kato, K., et al. (2018). Basic application of deep convolutional neural network to visual inspection. In *Proceedings of International Conference on Industrial Application Engineering (ICIAE2018)*, Okinawa (pp. 4–8).
4. Nagata, F., Tokuno, K., Otsuka, A., Ikeda, T., Ochi, H., Tamano, H., et al. (2018) Design tool of deep convolutional neural network for visual inspection. In *Proceedings of The Third International Conference on Data Mining and Big Data (DMBD2018), Springer-Nature LNCS Conference Proceedings 10943*, Shanghai (pp. 604–613).
5. Cristianini, N., & Shawe-Taylor, J. (2000) *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press.
6. Flores-Fuentes, W., Rivas-Lopez, M., Sergiyenko, O., Gonzalez-Navarro, F. F., Rivera-Castillo, J., Hernandez-Balbuena, D., et al. (2014). Combined application of power spectrum centroid and support vector machines for measurement improvement in optical scanning systems. *Signal Processing, 98*, 37–51.
7. Flores-Fuentes, W., Sergiyenko, O., Gonzalez-Navarro, F. F., Rivas-Lopez, M., Rodriguez-Quinonez, J. C., Hernandez-Balbuena, D., et al. (2016). Multivariate outlier mining and regression feedback for 3D measurement improvement in opto-mechanical system. *Optical and Quantum Electronics, 48*(8), 403.
8. Rodriguez-Quinonez, J. C., Sergiyenko, O., Hernandez-Balbuena, D., Rivas-Lopez, M., Flores-Fuentes, W., & Basaca-Preciado, L. C. (2014). Improve 3D laser scanner measurements accuracy using a FFBP neural network with Widrow-Hoff weight/bias learning function. *Opto-Electronics Review, 22*(4), 224–235.
9. Real, O. R., Castro-Toscano, M. J., Rodriguez-Quinonez, J. C., Serginyenko, O., Hernandez-Balbuena, D., Rivas-Lopez, M., et al. (2019). Surface measurement techniques in machine vision: Operation, applications, and trends. In *Optoelectronics in machine vision-based theories and applications* (pp. 79–104). Hershey: IGI Global.
10. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Lake Tahoe, NV (pp. 1097–1105).
11. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60*(6), 84–90.
12. Nagata, F., & Watanabe, K. (2002). Learning of contact motion using a neural network and its application for force control. In *Proceedings of the 4th Asian Control Conference (ASCC2002)* (pp. 420–424).
13. Nagata, F., Mizobuchi, T., Tani, S., Watanabe, K., Hase, T., & Haga, Z. (2009). Impedance model force control using neural networks-based effective stiffness estimator for a desktop NC machine tool. *Journal of Manufacturing Systems, 28*(2/3), 78–87.
14. Nagata, F., Mizobuchi, T., Hase, T., Haga, Z., Watanabe, K., & Habib, M. K. (2010). CAD/CAM-based force controller using a neural network-based effective stiffness estimator. *Artificial Life and Robotics, 15*(1), 101–105.
15. Nagata, F., & Watanabe, K. (2011). Adaptive learning with large variability of teaching signals for neural networks and its application to motion control of an industrial robot. *International Journal of Automation and Computing, 8*(1), 54–61.
16. Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14 (pp. 1–24).